



GENERAL  ELECTRIC

HOUSTON, TEXAS

TECHNICAL INFORMATION RELEASE

NASA CR-

160209

TIR 741-MED-3027

FROM	D. G. Fitzjerrell	TO	J. A. Rummel, Ph.D.
------	-------------------	----	---------------------

DATE 5/2/73	WORK ORDER REF: DM-110T	WORK STATEMENT PARA: NAS9-12932	REFERENCE:
----------------	----------------------------	------------------------------------	------------

SUBJECT

Program for Solution of Ordinary Differential Equations

(NASA-CR-160209) PROGRAM FOR SOLUTION OF
ORDINARY DIFFERENTIAL EQUATIONS (General
Electric Co.) 44 p HC A03/MF A01 . CSCL 09B

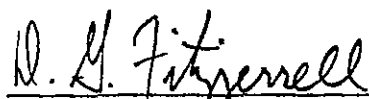
N79-25786

Unclas

G3/61 22198

This report is a description of, and user's instructions for a program for the solution of linear and nonlinear first-order ordinary differential equations. The program has a new integration algorithm for the solution of initial value problems which is particularly efficient for the solution of differential equations with a wide range of eigenvalues. The program in its present form will handle up to ten state variables, but is being expanded to handle up to fifty state variables.

Prepared by:


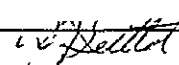

D. G. Fitzjerrell

Attachment
/db

REPRODUCED BY
NATIONAL TECHNICAL
INFORMATION SERVICE
U.S. DEPARTMENT OF COMMERCE
SPRINGFIELD, VA. 22161

CONCURRENCES

Counterpart:

Medical Projects  Engrg. & Advanced Programs
Unit Manager  Subsection Mgr. W.J. Beittel

DISTRIBUTION GE/AGS: Central Product File NASA/JSC: Technical Library/JM6
R. C. Croston, Ph.D. (1979 distribution)
V. J. Marks
R. F. Hassell

Page No.
1 of 1

NOTICE

THIS DOCUMENT HAS BEEN REPRODUCED FROM THE BEST COPY FURNISHED US BY THE SPONSORING AGENCY. ALTHOUGH IT IS RECOGNIZED THAT CERTAIN PORTIONS ARE ILLEGIBLE, IT IS BEING RELEASED IN THE INTEREST OF MAKING AVAILABLE AS MUCH INFORMATION AS POSSIBLE.

PROGRAM DESCRIPTION GUIDE

A. IDENTIFICATION

Program Name - Ordinary Differential Equation Solver - HMS

Author - H. Sloate, G.E. Co., Electronics Laboratory
Syracuse, N. Y.

Source Reference - An Implicit Formula for the Integration of
Stiff Network Equations, TIS No. R7OELS-2,
January 1970.

Programmer Contact - V. J. Marks, GE/MSc, Houston

Date of Issue - May 1, 1973

B. GENERAL DESCRIPTION

The program is designed to provide numerical solution to systems of linear and nonlinear first-order ordinary differential equations. The program contains a new integration algorithm for the solution of initial value problems and is particularly efficient for solving differential equations having a wide range of eigenvalues. For the classical methods, the integration step size is limited more by stability considerations than by accuracy. The new implicit fourth-order linear multistep method based on Gear's formula utilized in this program does not become numerically unstable as the time step size becomes large. Since the integration formula is a multistep method, it must be started by some other means. Gill's fourth-order Runge Kutta method is used to calculate the three points in addition to the initial conditions needed to start the multistep process. After the starting procedure is carried out and an acceptable time step size found, the step size is still kept small enough for accuracy. If it is too large, it is halved. If it is too small, it can cause excessive running times and an appropriate test is performed to detect this event and the step size is doubled.

C. USAGE AND RESTRICTIONS

Machine and Compiler Required - UNIVAC 1108 and FORTRAN

Peripheral Equipment Required - Card Reader; Line Printer

Approximate Amount of Memory - 6,027
Required

ORIGINAL PAGE IS
OF POOR QUALITY

D. PARTICULAR DESCRIPTION

Consider the system of first-order differential equations given in (1):

$$\dot{y} = f(y, t), \quad y(0) = y_0 \quad (1)$$

The function f may be a non-linear, time-varying function of y and hence it may be difficult or impossible to find an analytic solution to eq. (1) if a solution exists. If one does exist, the above system of equations may be integrated point by point in time to obtain an approximation to the solution of eq. (1), $y(t)$.

There are many formulas used for the integration of ordinary differential equations. Some are designed to integrate differential equations of high order¹ while others are used for a system of first-order differential equations such as (1).

The form given in eq. (1) is general because differential equations of any order can be rewritten as systems of first-order equations. Thus it is of interest to examine integration formulas which can be used to solve eq. (1).

Classical integration techniques include the linear multi-step methods, the predictor-corrector methods,² and the Runge Kutta methods.³ One might ask why new methods are being invented when we have such a repertoire of existing methods at our disposal. The following discussion will point out the shortcoming common to all classical methods.

1. STIFF DIFFERENTIAL EQUATIONS

A stiff set of linear ordinary differential equations is defined as one which has very large and very small eigenvalues. The term stiff probably comes from structural engineering where stiff members gave rise to large eigenvalues in the differential equation formulation.

When numerically integrating stiff differential equations, one encounters two conflicting requirements: a) the time step size must be large to reduce the number of steps taken and consequently the labor required to obtain the solution, and b) the time step must be small to prevent the integration algorithm from becoming numerically unstable. Stiff differential equations aggravate the above conflict because the maximum allowable time step for numerical stability is inversely proportional to the largest eigenvalue. Many time steps must be taken to display the solution associated with the small eigenvalues of the system. The combination of small step size and long running time causes the solution to be calculated at a great many points.

To see how a numerical integration formula can become unstable let us use Euler's forward integration formula given in eq. (2) to solve the linear first-order differential equation in eq. (3).

$$y_{n+1} = y_n + h\dot{y}_n \quad (2)$$

$$\dot{y} = -\lambda y, \quad y(0) = y_0 \quad (3)$$

Substituting eq. (3) into (2) and solving we obtain the recursion relationship:

$$y_{n+1} = (1 - \lambda h)y_n \quad (4)$$

The solution to eq. (4) is

$$y_n = (1 - \lambda h)^n y_0 \quad (5)$$

For the sequence $[Y_n]$ to be bounded it follows that:

$$\begin{aligned} |1 - \lambda h| &\leq 1 \\ 0 \leq \lambda h &\leq 2 \\ 0 \leq h &\leq \frac{2}{\lambda} \end{aligned} \quad (6)$$

That is, if the numerical solution is to have the same property as the actual solution to eq. (3) (i. e., boundedness) then there is an upper limit on h which is inversely proportional to the largest eigenvalue in the system of equations being integrated. This is generally true of all classical integration formulas.

2. IMPLICIT INTEGRATION FORMULAS

Much research has been done in recent years to circumvent the numerical instability problem. One approach is to examine the equations being integrated and to modify them to delete the troublesome eigenvalues.^{4, 5} However, this may be time consuming and not completely general. Another approach is to use a non-linear integration formula which uses exponentials rather than polynomials to fit the functions being integrated.⁶ This type of formula has excellent stability properties and is exact for linear systems (no truncation error). However, the Jacobian of the system of differential equations must be calculated and the exponential matrix generated. Here we have solved the numerical instability problem but have replaced it with a great deal of computation per time step. A tradeoff between stability and computation per time step can be made by using the convergence properties of the exponential power series as is done in the CIRCUS⁷ network analysis program.

Lacking a complete solution to the problem, let us turn our attention to linear multi-step integration formulas of the form:

$$y_{n+1} = \sum_{i=0}^{k-1} \alpha_i y_{n-i} + h \sum_{i=-1}^{k-1} \beta_i \dot{y}_{n-i} \quad (7)$$

The α 's and β 's may be determined by requiring that the formula be exact up to a certain order, p , if the function, $y(t)$, being integrated is a polynomial in time of order p or less. If there are more α 's and β 's than necessary to make the formula exact up to and including order p , then the remaining arbitrary α 's and β 's may be chosen to optimize other properties of the integration formula.²

If $\beta_{-1} \neq 0$ the integration formula is said to be an implicit formula because the unknown, y_{n+1} , appears on both sides of the equation. These equations have the same form as the corrector formulas in the predictor-corrector method. If $\beta_{-1} = 0$ the integration formula is an explicit formula, similar to the predictor formulas in the predictor-corrector method.

The implicit equations can have numerical stability for the time step size, h , arbitrarily large, as the following example shows.⁸ Consider Euler's backward formula which is an implicit formula of order one and step number $k = 1$.

$$y_{n+1} = y_n + h \dot{y}_{n+1} \quad (8)$$

If we use eq. (8) to solve eq. (3) we obtain the difference equation:

$$y_{n+1} = (1 + \lambda h)^{-1} y_n \quad (9)$$

The solution to eq. (9) is

$$y_n = (1 + \lambda h)^{-n} y_0 \quad (10)$$

If $\text{Re}[\lambda] \geq 0$, then $[y_n]$ is a bounded sequence for all $h \geq 0$. Thus there is no upper limit on time step size due to numerical instability problems. Here is one formula, at least, which has the desired numerical stability properties.

The question arises as to whether there are more formulas with the above stability properties. Before answering this question let us present the concept of A-stability.⁹ A k step method is called A-stable if all solutions of eq. (7) tend to zero, as $n \rightarrow \infty$, when the method is applied with fixed positive h to eq. (3) where λ is a complex constant with positive real part. A-stability requires that if the solution to the differential equation is stable then the corresponding solution to the difference equation be stable also. The example given in eq. (8) is an A-stable method. Unfortunately there are not too many more of them. Dahlquist⁹ proved two theorems important to our discussion:

Th. 1. An explicit linear multistep method cannot be A-stable.

Th. 2. The order, p , of an A-stable linear multistep method cannot exceed 2.

Theorem 2 indicates that if we require A-stability we will be able to use only formulas of low order and hence limited accuracy. If the restriction of A-stability is removed, we can find high-order formulas with acceptable numerical stability properties.

In Appendix I it is shown that if we want an integration formula which remains stable as $h \rightarrow \infty$, we must use an implicit formula and its maximum order cannot exceed its step number k . Such an integration formula will not be A-stable, however, if $p > 2$.

3. INTEGRATION FORMULA SELECTION

As $h \rightarrow \infty$ it would be good to have the roots of the integration formula tend to zero so that the unimportant large eigenvalue modes of a system of differential equations would rapidly die out as the time step size is increased. This is the approach taken by Gear¹⁰ in obtaining implicit integration formulas of order 2 through 6. In eq. (7) there are $2k+1$ arbitrary constants. If a k th order fit is required, $k+1$ of the constants are fixed. If the k roots of $\sum_{i=0}^k \beta_i \zeta^i = 0$ are all to be zero, the remaining k constants of eq. (7) are fixed and the integration formula is unique. The formulas of order 2 through 6 are given in Appendix II. Generally speaking it is desirable to use as high an order formula as possible to achieve greater accuracy. The higher order formulas have the disadvantage in that more past values must be stored and more manipulation is required when

the step size is halved or doubled. A fourth-order formula was chosen for integrating the systems of differential equations as a compromise between accuracy and complexity. It is different from Gear's fourth-order formula in that its coefficients were selected to achieve a compromise between truncation error and roots close to zero as $h \rightarrow \infty$. This was done by minimizing a performance index using the Fletcher Powell method.^{20, 21} The performance index was a weighted sum of the truncation error given in eq. (12) and the square of the sum of the squares of the roots of eq. (7) as $h \rightarrow \infty$.

4. TRUNCATION ERROR

The truncation error of the integration formula in eq. (7) is due to the fact that the formula has only a finite number of terms. This error would be present even if there were no roundoff errors in the computer due to using numbers with a finite number of bits. An expression for the truncation error can be obtained by integration by parts.¹¹ (See Appendix III.) It is given by

$$T_n = \frac{y^{(k+1)}(N)}{k!} \int_{(n-k+1)h}^{(n+1)h} G(s) ds \quad (11)$$

where $G(s)$ is called the influence function and is of the same sign over the interval from $(n+1-k)h$ to $(n+1)h$. The $k+1$ th derivative is evaluated at N which is somewhere in the interval from $(n-k+1)h$ to $(n+1)h$. If $G(s)$ changes sign over the interval, the First Mean Value Theorem¹² which was used in obtaining eq. (11) does not apply and an estimate of the error can be obtained from:

$$T_n = \frac{|y^{(k+1)}(N)|}{k!} \int_{(n-k+1)h}^{(n+1)h} |G(s)| ds \quad (12)$$

Equations (11) and (12) were used to obtain the truncation error of the integration formulas studied.

If $G(s)$ is of the same sign over the interval and eq. (11) applies, an alternate shorter method can be used to compute T_n . Assume the error is given by

$$T_n = \frac{E_k}{(k+1)!} h^{k+1} y^{(k+1)}(N) \quad (11a)$$

Assume $Y(t) = t^{k+1}$ and substitute into eq. (7). Since eq. (11a) is the error in eq. (7) E_k can be evaluated. This is the standard way to find error terms in quadrature formulas. Note, however, that it is valid only if $G(s)$ is of the same sign over the interval covered by the integration formula. (See Appendix III for an example of the procedure for calculating E_k .)

5. STABILITY BOUNDARIES

Let $\dot{y} = qy$. The roots of the difference equation given in eq. (7) are the zeros of

$$\sum_{i=0}^{k-1} (\alpha_i + qh\beta_i) \zeta^{k-1-i} = 0, \quad \alpha_{-1} = -1 \quad (13)$$

Solving for qh gives

$$qh = \frac{\sum_{i=0}^{k-1} \alpha_i \zeta^{k-1-i}}{\sum_{i=-1} \beta_i \zeta^{k-1-i}} \quad (14)$$

If there is some value of qh which causes ζ to have unit magnitude, then we can let $\zeta = e^{j\theta}$ and let θ vary from 0 to π . This will map the corresponding stability boundary in the qh plane. If the stability boundary lies wholly in the plane $\text{Re}[qh] \geq 0$, then the integration formula being considered will be A-stable. If not, then there are some values of qh in the plane $\text{Re}[qh] < 0$ for which the roots are not less than one in magnitude and by definition the method is not A-stable.

6. COMPARISON OF TWO FOURTH-ORDER FORMULAS

Gear's fourth-order formula has a truncation error given by:

$$|T_n| = \frac{2.304}{4!} h^5 |y^{(5)}(\eta)| \quad (15)$$

The truncation error of the optimized fourth-order formula given in Appendix II is:

$$|T_n| = \frac{1.932}{4!} h^5 |y^{(5)}(\eta)| \quad (16)$$

These expressions can be obtained by integrating the influence functions shown in Figure 1. See Appendix III.

The stability boundaries in the qh plane of are shown in Figures 2 and 3. Both methods are $A(\alpha)$ stable in the sense of Widlund.¹³ Gear's method has an $\alpha = 72$ degrees. The optimized method has an $\alpha = 63$ degrees. It can be seen that there is a trade-off between truncation error and the size of the stability sector in the qh plane. The optimized formula is more accurate than Gear's method but its stability sector is smaller.

7. SOLUTION OF THE IMPLICIT EQUATION

The integration formula given in eq. (7) is implicit if $\beta_{-1} \neq 0$. Since the unknown, y_{n+1} , appears on both sides of the equal sign, the equation will have to be solved iteratively. One approach which is used in the predictor-corrector methods is the method of successive substitution (Picard's Method). The k^{th} estimate of y_{n+1} is used on the right side of eq. (7) to evaluate \dot{y}_{n+1} , and this is used to evaluate the $k+1$ th estimate of y_{n+1} .

$$y_{n+1}^{(k+1)} = y_{n+1}^{(k)} + h \sum_{i=0}^{k-1} (\alpha_i y_{n-i}^{(k)} + h \beta_i \dot{y}_{n-i}^{(k)}) \quad (17)$$

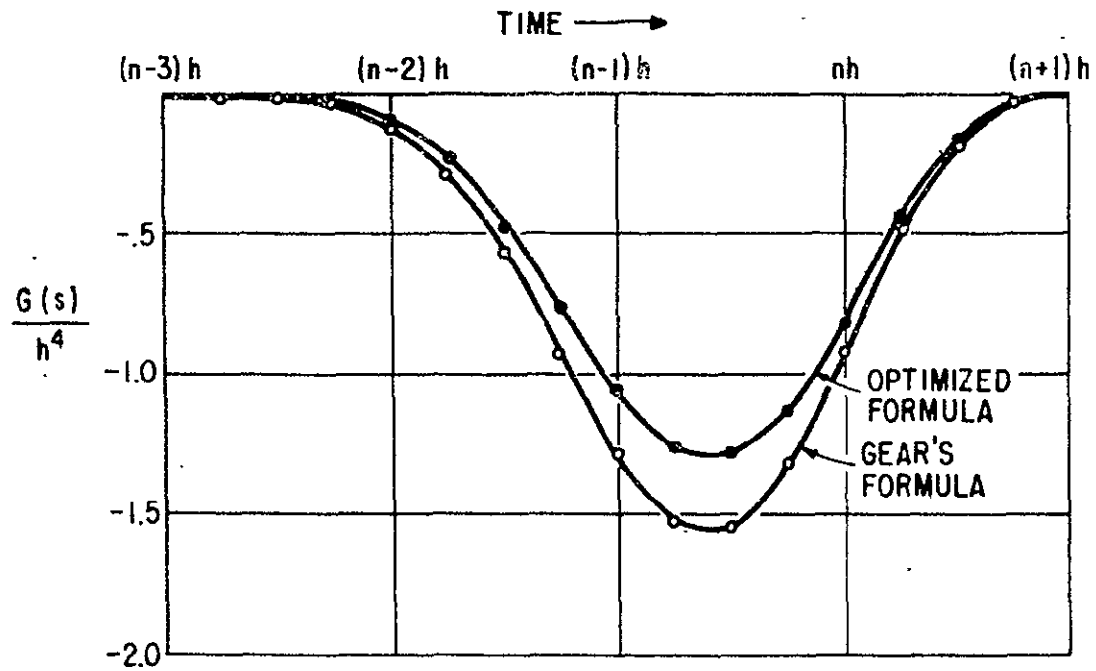


Figure 1. Influence Function for Two Fourth Order Formulas

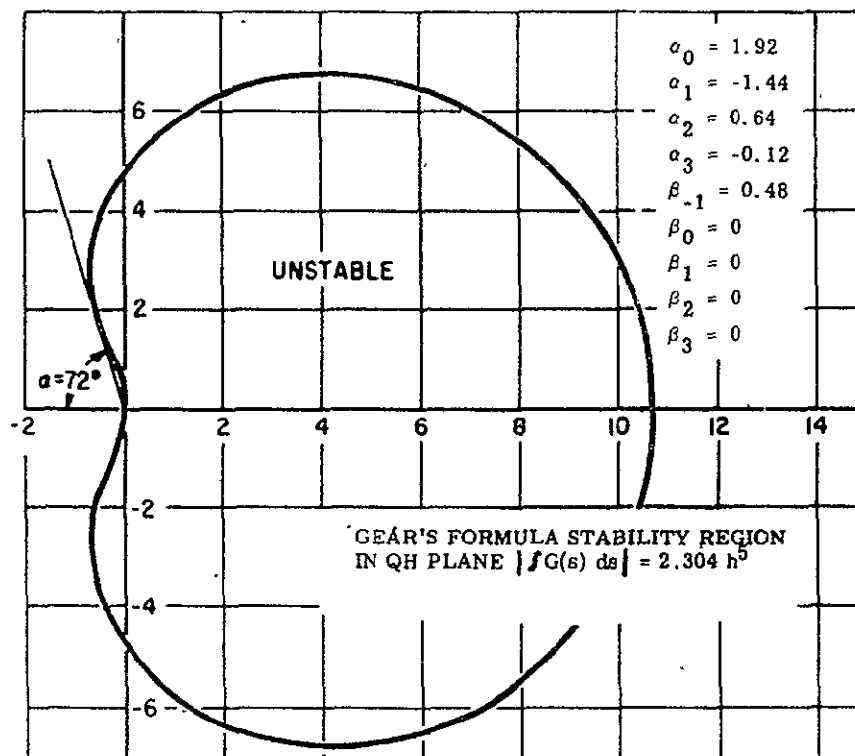


Figure 2. Stability Region for Gear's Formula

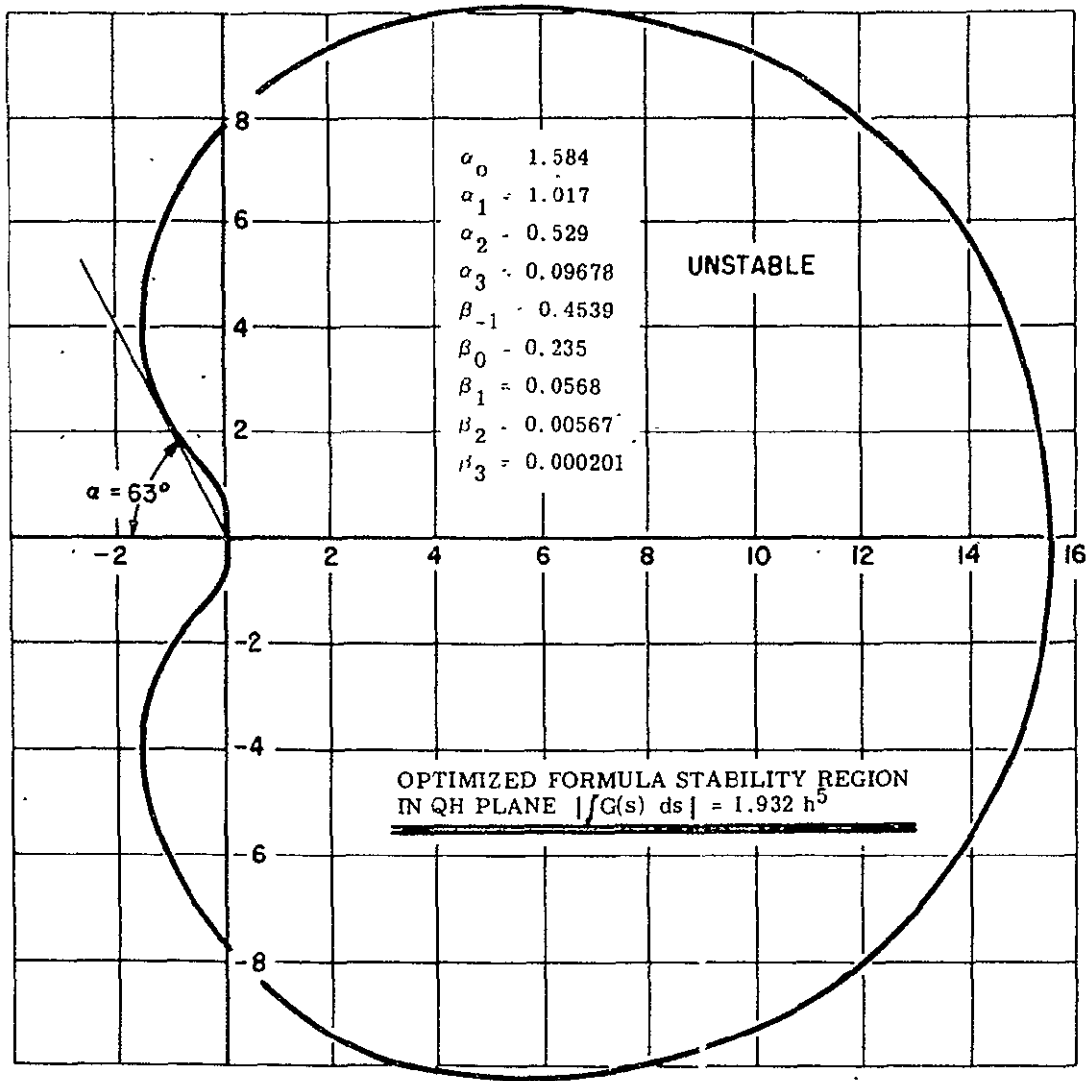


Figure 3. Stability Region for Optimized Formula

To see if this can be of use let us examine the linear case $\dot{y} = qy$. Letting

$$K = \sum_{i=0}^{k-1} \alpha_i y_{n-i} + h\beta_i y_{n-i}$$

because K does not change during the iteration, we find that eq. (17) is of the form:

$$y_{n+1}^{(k+1)} = h\beta_{-1}q y_{n+1}^{(k)} + K \quad (18)$$

If $y_{n+1}^{(0)}$ is the initial guess for y_{n+1} , the solution to eq. (18) is:

$$y_{n+1}^{(k)} = (qh)^k y_{n+1}^{(0)} + \frac{1 - (qh)^k}{1 - qh} K \quad (19)$$

This is a divergent sequence if $|qh| > 1$. This is the same restriction we were trying to avoid by using implicit methods in the first place. It can be shown that this restriction holds for systems of equations and nonlinear equations as well.⁸

The Newton Raphson method can be used to solve implicit equations such as eq. (7). It does not limit the size of qh . If eq. (7) is applied to systems of equations such as:

$$\begin{aligned} Q(y_{n+1}) &= y_{n+1} - \beta_{-1} h \dot{y}_{n+1} \\ &\quad - \sum_{i=0}^{k-1} (\alpha_i y_{n-i} + h \beta_i \dot{y}_{n-i}) \end{aligned} \quad (20)$$

it can be seen that $Q(y_{n+1})$ will be zero at the solution. If Q is expanded in a Taylor series and only the first term is kept, we have

$$Q(y_{n+1}^{(k)} + \Delta y_{n+1}) = Q(y_{n+1}^{(k)}) + J \cdot \Delta y_{n+1}$$

Solving eq. (21) for Δy_{n+1} gives

$$\Delta y_{n+1} = -J^{-1} Q(y_{n+1}^{(k)}) \quad (22)$$

where J is the Jacobian of Q . The Jacobian can be found from eq. (20) and is

$$J = I - \beta_{-1} h A \quad (23)$$

where A is given by

$$\left[\frac{\partial f_i}{\partial y_j} \right], \quad \begin{aligned} \dot{y}_1 &= f_1 \\ \dot{y}_2 &= f_2 \\ &\vdots \\ \dot{y}_n &= f_n \end{aligned}$$

If the differential equations being integrated are linear, the matrix A does not change during the problem and needs to be calculated only once. For very nonlinear equations A may change rapidly and a new Jacobian may have to be calculated at each time step. In practice, however, a new Jacobian is not calculated this often. If the number of iterations required for the Newton method to converge exceeds a certain number, the Jacobian is recalculated using difference techniques, it is inverted, and a new Δy_{n+1} is found. If the number of iterations per step does not exceed the limit, the inverse Jacobian from the previous step is used. In this manner considerable time is saved which would be used in evaluating the derivatives \dot{y}_{n+1} .

If after calculating a new Jacobian the Newton method still fails to converge, the step size h is halved and eq. (7) is applied once more. If after h has been halved a certain number of times there is still no convergence, the simulation is terminated and an error message printed out.

If the system of differential equations being integrated is very stiff and the transient has died out, $qh \gg 1$ for all the eigenvalues of the system.

$$[I - \beta_{-1} h A]^{-1} \approx -\frac{1}{\beta_{-1} h} A^{-1} \quad (24)$$

Equation (24) shows that inverting the Jacobian is almost the same as inverting A which has extreme ranges in eigenvalues. Due to roundoff in the computer, A seems nearly singular and gives very inaccurate results when it is inverted. An iterative scheme to correct the elements of the inverse matrix has been used to overcome the problem.¹⁴ It greatly reduces the number of iterations used for convergence of Newton's method in some cases.

8. SIMULATION PROGRAM ORGANIZATION AND LOGIC

a. Starting Procedure

Since the integration formula in eq. (7) is a multistep method it must be started by some other means. Gill's¹⁵ fourth-order Runge Kutta method was chosen to calculate the three points in addition to the initial conditions needed to start the multistep method. Two more points are calculated using the multistep method and these six points are used to approximate the fifth derivative of $y(t)$ by difference techniques. From this the truncation error is calculated and the accuracy test is made to determine if the time step size is too large. If it is, the step size is reduced by a factor of ten and the starting procedure is re-initialized. The error test is:

$$|T_n| < (1 + |y_{n+1}|) * ELIM \quad (25)$$

If this inequality is satisfied, the time step size is acceptable. $ELIM$ is data input by the user. If $y(t)$ is an extremely large number, roundoff error in the computer will keep T_n from being less than $ELIM$. The $|y_{n+1}|$ term is included to prevent the occurrence of this situation.

b. Time Step Size Control

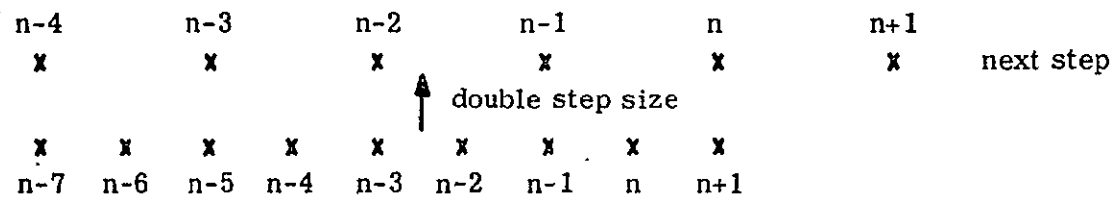
After the starting procedure has been successfully carried out and an acceptable time step size, h , found, eq. (25) is still used to keep h small enough for accuracy. If it is too large, it is halved. If it is too small, it can cause excessive running times. Hence the test in eq. (26) is made.

$$|T_n| < (1 + |y_{n+1}|) * ELIM/40. \quad (26)$$

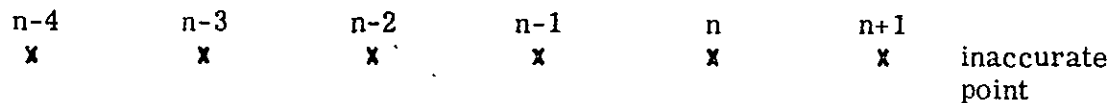
If this inequality is satisfied, the step size is doubled. The factor of 40. was chosen because the error is proportional to h^5 . Doubling h multiplies the truncation error by 32 if $y^{(5)}(\eta)$ remains constant. The factor of 40 was chosen so that eq. (25) would still be satisfied after h was doubled.

Since six points are necessary to calculate the fifth derivative, it is necessary to have five accurate past points before y_{n+1} is calculated. Before doubling h , eight past values of $y(t)$ are required so that there will be five past values of $y(t)$ when y_{n+1} is calculated on the next step. (See Figure 4.) The step-size doubling is inhibited until eight past points have been calculated.

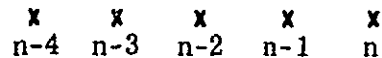
If y_{n+1} is calculated and the truncation error is too large, the step size is halved. (See Figure 4.) y_{n-1} and y_{n-3} are found by interpolation. \dot{y}_{n-1} and \dot{y}_{n-3} are calculated from these values using the state equations. The old y_{n-1} , \dot{y}_{n-1} are placed in y_{n-2} , \dot{y}_{n-2} . The old y_{n-2} , \dot{y}_{n-2} are placed in y_{n-4} , \dot{y}_{n-4} . A new y_{n+1} is calculated from the integration formula. If the truncation error is still too large, the step size is halved again and the process repeated.



DOUBLING THE STEP SIZE



↓ halve step size



HALVING THE STEP SIZE

Figure 4. Logic for Changing Step Size

c. The Newton Iteration

The Newton Raphson method is used to drive Q to zero. An initial estimate of the solution, $y_{n+1}^{(0)}$, is calculated by extrapolation using the five previous points which have been stored. Using this estimate $Q^{(0)}$ is calculated and Δy_{n+1} is calculated.

$$y_{n+1}^{(k+1)} = y_{n+1}^{(k)} + TI * \Delta y_{n+1} \quad (27)$$

The new value of $y_{n+1}^{(1)}$ is computed using eq.(27) with $TI = 1$. $Q^{(1)}$ is evaluated and if $Q^{(1)T} Q^{(1)} < Q^{(0)T} Q^{(0)}$, a new Δy_{n+1} is calculated and the process is repeated. If not, TI is decreased by a factor of 10 and eq. (27) is applied again with the same Δy_{n+1} . This process is repeated up to NDEX times. If there is still no success, a new Jacobian is calculated and a new Δy_{n+1} is used in eq. (27). If there is no success even then, h is halved and the whole procedure is tried again. h can be halved up to NHLIM times. If no success is obtained, an error message is printed and the program terminated.

New Δy_{n+1} values are checked in the inequality

$$|\Delta y| < (SC + |y|) * XNLIM \quad (28)$$

If eq. (28) is satisfied for all the state variables, the iteration has been completed successfully and the last estimate of y_{n+1} is taken to be the solution of the implicit integration formula. SC is usually set to 1. and $XNLIM$ is usually set to $0.1 * ELIM$.

Each time a Δy_{n+1} step has been successfully carried out, the estimate of the inverse Jacobian is updated using Broyden's scheme.¹⁶ This updating keeps the estimate of the inverse Jacobian current and lessens the need for calculating new Jacobians.

F. DESCRIPTION OF INPUT

The differential equation solver is a stand-alone program. Flow charts of the program are shown in Appendix IV. It needs two sets of input from the user. The first is input data which is read in on three cards.

The first card has a format (2F8.1, I5) and reads the variables TF , PT , NV . TF is the final time for the problem. When the solution time reaches this value, execution will terminate. PT is the number of points in time the user wants printed out. For example, if $TF = 10.$ and $PT = 5.$, the state variables would be printed out at $t = 2., 4., 6., 8.,$ and $10.$ seconds. NV is the number of state variables.

The second data card has a format (10F8.2). The initial values of the state variables (up to 10) are input here. If there are less than 10 state variables, leave the unused columns blank.

The third card has a format (4E7.1) and reads the variables $YLIM$, $ELIM$, $XNLIM$, SC . $YLIM$ is usually set at about 10^{-5} and governs how the state variables are perturbed to calculate the elements in the Jacobian matrix. The program uses $YLIM$ as follows:

$$\begin{aligned} \text{If } |y_n| < YLIM, \text{ perturb } y_n \text{ by } 10^{-4}. \\ \text{If } |y_n| < YLIM, \text{ perturb } y_n \text{ by } 10^{-3} * y_n \end{aligned}$$

$ELIM$ controls the truncation error of the integration formula. See eqs. (25) and (26). It is usually set at about 10^{-3} . If a more accurate solution is required, it can be set smaller. However, a smaller $ELIM$ requires more time steps to complete a simulation. $XNLIM$ controls the convergence requirements for the Newton iteration. See eq.(28). It is usually set to $0.1 * ELIM$.

The other input required of the user is a subroutine in FORTRAN which describes the system of equations being integrated. An example is shown below. The maximum number of state variables is 10. The state variables are contained in the array X. The first derivatives of X are stored in the array D. The variable NUM is used to count how many times the state equations have been evaluated during a simulation.

```

1      CSTATE
2      SUBROUTINE STATE(X,D)
3      COMMON Y0,DY0,DB,T,H,NV,NUM
4      DOUBLE PRECISION Y0(10),DY0(10),DB(10)
5      DOUBLE PRECISION X(1),D(1)
6      D(2)=X(1)-X(2)
7      D(1)=-1000.*(X(1) + D(2))
8      NUM = NUM+1
9      RETURN
10     END

```

F. DESCRIPTION OF OUTPUT

Since the integration formula is a variable step method, the variables may not be calculated at the values of time at which printout is desired. Hence, it is necessary to use Lagrangian¹⁷ interpolation to find the values of the state variables at the desired time. See Appendix B for Sample Output.

G. INTERNAL CHECKS AND EXITS

None.

H. INDEPENDENT SUBROUTINES

None

I. SYSTEM SUBROUTINES

No special subroutine.

J. COMPLETION OR FINAL CHECKOUT DATE

April 25, 1973

Source listing and sample cases are given in Appendix B.

BIBLIOGRAPHY

1. Ralston, A. A First Course In Numerical Analysis, McGraw-Hill, New York, 1965, pp. 210-213.
2. Hamming, R.W. Numerical Methods For Scientists and Engineers, McGraw-Hill, New York, 1962, pp. 165-210.
3. Kopal, Z. Numerical Analysis, Chapman and Hall, London, 1955.
4. Brayton, R.K., Gustavson, F.G., Liniger, W. "A Numerical Analysis of the Transient Behavior of a Transistor Circuit," IBM Journal of Research and Development, Vol. 10 (July, 1966), pp. 292-299.
5. Shell, D.L. "Extending the Stability of AMSINT Integration System," G.E. Technical Information Series, Report Number 63GL32, Schenectady, N.Y., January, 1963.
6. Pope, D.A. "An Exponential Method of Numerical Integration of Ordinary Differential Equations," Communications of the ACM, Vol. 6 (August 1963), pp. 491-493.
7. Milliman, L.D., Massena, W.A., and Dickhaut, R.A. CIRCUS User's Guide, U.S. Army Materiel Command, Harry Diamond Laboratories, Washington, D.C. 20438
8. Sandberg, I.W. and Shichman, H. "Numerical Integration of Systems of Stiff Nonlinear Differential Equations," Bell System Technical Journal, Vol. 47 (April, 1968), pp. 511-527.
9. Dahlquist, G. "A Special Stability Problem For Linear Multistep Methods," BIT 3, (1963), pp. 27-43.
10. Gear, C.W. "The Automatic Integration of Stiff Ordinary Differential Equations," IFIPS Congress, (August 1968), Booklet A, pp. A81-A95. or Gear, C.W. "Numerical Integration of Stiff Ordinary Differential Equations," Technical Report No. 221, Dept. of Computer Science, Univ. of Illinois, (January 1967).
11. Ralston, A. A First Course In Numerical Analysis, McGraw-Hill, New York, 1965, pp. 165-168.
12. Rudin, W. Principles of Mathematical Analysis, McGraw-Hill, New York, 1964, p. 123.
13. Widlund, O.B. "A Note on Unconditionally Stable Linear Multistep Methods," BIT 7, (1967), pp. 65-70.
14. Faddeev, D.K. and Faddeeva, V.N. Computational Methods of Linear Algebra, W.H. Freeman Co., San Francisco, 1963, pp. 159-161.
15. Gill, S. "A Process for the Step-By-Step Integration of Differential Equations in an Automatic Digital Computing Machine," Proc. Cambridge Philos. Soc., Vol. 47, (1951), pp. 96-108.
16. Broyden, C.G. "A Class of Methods for Solving Nonlinear Simultaneous Equations," Mathematics of Computation, Vol. 19, No. 92, (Oct. 1965), pp. 577-593.
17. Ralston, A. A First Course In Numerical Analysis, McGraw-Hill, New York, 1965, pp. 42-44.
18. Hamming, R.W. Numerical Methods for Scientists and Engineers, McGraw-Hill, New York, 1962, pp. 177 and 203.

19. IBM System/360 Scientific Subroutine Package, Publication Number H20-0205-3, pp. 337-343.
20. Fletcher, R. and Powell, M.J.D. "A Rapidly Convergent Descent Method for Minimization," Computer Journal, Vol. 6, Issue 2, (1963), pp. 163-168.
21. IBM System/360 Scientific Subroutine Package, Publication Number H20-0205-3, pp. 221-225.

APPENDIX A

MATHEMATICAL PROOFS

GEAR'S FORMULA

TRUNCATION ERROR

MATHEMATICAL PROOFS

DEFINITION - A linear multistep method is said to be consistent if it is of at least order one.

If a linear multistep method is consistent, $\sum_{i=0}^k \alpha_i \zeta^i$ has a simple root at $\zeta = 1$. Let $y = K$, a constant. $\therefore \dot{y} = 0$. Substituting this into

$$\sum_{i=0}^k \alpha_i y_{n+1} - h \sum_{i=0}^k \beta_i \dot{y}_{n+i} = 0 \quad (1)$$

results in

$$\sum_{i=0}^k \alpha_i = 0 \quad (2)$$

If there are two roots at $\zeta = 1$, the solution of the difference equation is

$$y_n = K_1 + K_2 n + K_3 \zeta_3^n + \dots + K_k \zeta_k^n \quad (3)$$

where the K 's are determined by initial conditions. The sequence $[y_n]$ is not convergent as $n \rightarrow \infty$ because of the term containing n . Hence the method will

not be convergent if $\sum_{i=0}^k \alpha_i \zeta^i$ has more than one zero at $\zeta = 1$. (See Henrici, P.

Discrete Variable Methods in Ordinary Differential Equations, Wiley, New York, 1962, pp. 217-218.)

We will now show that consistency requires that $\sum_{i=0}^k \beta_i \neq 0$. Let $\dot{y} = 1$,

$y(0) = 0$. The solution to the differential equation is $y(t) = t$. Equation (1) becomes

$$\sum_{i=0}^k \alpha_i \cdot i - \sum_{i=0}^k \beta_i = 0 \quad (4)$$

Since $\frac{d}{d\zeta} \left[\sum_{i=0}^k \alpha_i \zeta^i \right] \neq 0$ because $\zeta = 1$ is a simple root, it follows from

eq. (4) that $\sum_{i=0}^k \beta_i \neq 0$.

Lemma 1. If the following three hypotheses are true:

- 1) $\frac{1}{\sum_{i=0}^{\infty} b_i Z^i} = \sum_{i=0}^{\infty} a_i Z^i$
- 2) $b_0 = 1, b_i > 0 \quad i = 1, 2, \dots$
- 3) $b_{n+1} b_{n-1} - b_n^2 > 0$

then $a_0 = 1$ and $a_i < 0 \quad i = 1, 2, \dots$

Proof:

If $\sum_{i=0}^{\infty} b_i Z^i$ has a non-zero radius of convergence, then $\sum_{i=0}^{\infty} a_i Z^i$ does also.

(See Knopp, K. Infinite Sequences and Series, Dover, 1956, p. 116.)

From

$$\sum_{i=0}^{\infty} b_i Z^i \cdot \sum_{i=0}^{\infty} a_i Z^i = 1$$

we obtain the following:

$$\begin{aligned} b_0 a_0 &= a_0 = 1 \\ b_1 + a_1 &= 0 \\ b_2 + b_1 a_1 + a_2 &= 0 \\ &\vdots \\ b_n + b_{n-1} a_1 + \dots + b_1 a_{n-1} + a_n &= 0 \end{aligned}$$

Solving for b_n :

$$b_n = - \sum_{j=1}^n b_{n-j} a_j \quad (5)$$

In a similar manner

$$b_{n+1} = - \sum_{j=1}^n b_{n+1-j} a_j - a_{n+1} \quad (6)$$

Multiplying eq. (6) by b_n and eq. (5) by b_{n+1} and subtracting gives

$$b_n a_{n+1} = \sum_{j=1}^n (b_{n-j} b_{n+1} - b_{n+1-j} b_n) a_j \quad (7)$$

Since by hypothesis $b_{n+1} b_{n-1} - b_n^2 > 0$

$$\frac{b_0}{b_1} > \frac{b_1}{b_2} > \frac{b_2}{b_3} > \dots > \frac{b_{n-j}}{b_{n+1-j}} > \dots > \frac{b_n}{b_{n+1}}$$

$b_{n-j} b_{n+1} - b_{n+1-j} b_n > 0$ and this shows the term in parenthesis in eq. (7) is always positive. Equation (7) will now be used to show by induction that $a_i < 0$, $i = 1, 2, \dots$

$$a_0 = 1$$

$$a_1 = -b_1 < 0$$

from eq. (7)

$$b_1 a_2 = (b_0 b_2 - b_1^2) a_1 < 0$$

$$\therefore a_2 < 0$$

Assume $a_1, a_2, \dots, a_n < 0$. From eq. (7)

$$a_{n+1} = \frac{1}{b_n} \sum_{j=1}^n (b_{n-j} b_{n+1} - b_{n+1-j} b_n) a_j$$

Since all the terms in parentheses are positive, b_n is positive, and the a_j 's are negative, a_{n+1} must be negative. This completes the induction.

Theorem: No explicit linear multistep method remains stable as h becomes arbitrarily large. $p = k$ is the highest order possible for an implicit linear multistep method which remains stable as h becomes arbitrarily large.

Proof: let $\dot{y} = qh$ (8)

The linear multistep method used to solve eq. (8) can be written as:

$$\sum_{i=0}^k \alpha_i y_{n+i} - h \sum_{i=0}^k \beta_i \dot{y}_{n+i} = O(h^{p+1}) \quad (9)$$

Substituting the solution $y = e^{qt}$ of eq. (8) into eq. (9):

$$\sum_{i=0}^k \left[\alpha_i (e^{qh})^{n+i} - qh \beta_i (e^{qh})^{n+i} \right] = O(h^{p+1}) \quad (10)$$

Let us use the transformation $\zeta = e^{qh}$ which maps the LHP of qh into the unit circle in the ζ plane. Dividing eq. (10) by qh and substituting $\zeta = e^{qh}$ and $qh = \ln \zeta$:

$$\sum_{i=0}^k \frac{\alpha_i \left(\frac{1-Z}{1-\zeta} \right)^i}{\ln \left(\frac{1+Z}{1-\zeta} \right)} - \sum_{i=0}^k \beta_i \zeta^i = O(|\ln \zeta|^p) \quad (11)$$

The ζ 's may be viewed as the roots of eq. (9). For stability they must remain in the unit circle. The transformation $Z = \frac{\zeta - 1}{\zeta + 1}$ maps the interior of the unit circle into the LHP of the Z plane. The inverse relationship is

$$\zeta = \frac{1 + Z}{1 - Z} \quad (12)$$

Substituting eq. (12) into eq. (11)

$$\frac{\sum_{i=0}^k \alpha_i \left(\frac{1+Z}{1-Z}\right)^i}{\ln \left(\frac{1+Z}{1-Z}\right)} - \sum_{i=0}^k \beta_i \left(\frac{1+Z}{1-Z}\right)^i = O\left(\left[\ln \left(\frac{1+Z}{1-Z}\right)\right]^p\right) \quad (13)$$

As $h \rightarrow 0$, $\zeta \rightarrow 1$, and $Z \rightarrow 0$. Therefore multiplying eq. (13) by $\left(\frac{1-Z}{2}\right)^k$ and simplifying:

$$\frac{r(Z)}{\ln \left(\frac{1+Z}{1-Z}\right)} - S(Z) = O(Z^p) \quad (14)$$

where

$$r(Z) = \left(\frac{1-Z}{2}\right)^k \sum_{i=0}^k \alpha_i \left(\frac{1+Z}{1-Z}\right)^i = \sum_{i=0}^k a_i Z^i \quad (15)$$

$$S(Z) = \left(\frac{1-Z}{2}\right)^k \sum_{i=0}^k \beta_i \left(\frac{1+Z}{1-Z}\right)^i = \sum_{i=0}^k b_i Z^i \quad (16)$$

If eq. (14) is to be of order p then the coefficients on the left side of eq. (14) must cancel up to the Z^p term. $r(Z)$ has a zero at $Z = 0$ because $\sum_{i=0}^k \alpha_i \zeta^i$ has a root at $\zeta = 1$. Since eq. (9) is stable for $h = 0$ all the roots of $\sum_{i=0}^k \alpha_i \zeta^i$ must be on or within the unit circle and hence $a_i \geq 0$ in eq. (15). If we desire that all the roots of eq. (9) be on or within the unit circle as $h \rightarrow \infty$ then $b_i \geq 0$ also. Let

$$\frac{Z}{\ln \frac{1+Z}{1-Z}} = C_0 + C_2 Z^2 + C_4 Z^4 + \dots \quad (17)$$

Due to Lemma 1:

$$C_0 > 0, C_{2\nu} < 0, \nu = 1, 2, \dots$$

Matching coefficients in eq. (14) gives

$$b_0 = c_0 a_1$$

$$b_1 = c_0 a_2$$

(18)

$$b_{2\nu} = c_0 a_{2\nu+1} + c_2 a_{2\nu-1} + \dots + c_{2\nu} a_1$$

$$b_{2\nu+1} = c_0 a_{2\nu+2} + c_2 a_{2\nu} + \dots + c_{2\nu} a_2$$

Since $S(Z)$ is a polynomial of k th order $b_n = 0$, $n > k$. The same applies for $a_n = 0$, $n > k$.

Assume k is odd.

$$b_{2\nu+1} = b_k = c_0 a_{k+1} + c_2 a_{k-1} + \dots + c_k a_2$$

Since $a_{k+1} = 0$, the coefficient matching in eq. (14) requires that $b_k < 0$. But by Descartes's Rule of Signs this means that $S(Z)$ has roots in the right half of the Z plane and hence $|\zeta| > 1$ as $h \rightarrow \infty$. The conclusion is that we can only require coefficient matching in eq. (14) up to the $k-1$ th term. Hence $p = k$. The same reasoning holds if k is even.

If eq. (9) is explicit, then $\beta_k = 0$. From eq. (16) $S(1) = 0 = \sum_{i=0}^k b_i$. Since $b_i \geq 0$, $i = 0, \dots, k$ if all the roots are to be within the unit circle as $h \rightarrow \infty$, all the b_i 's must be zero, including b_0 . But $b_0 = \left(\frac{1}{2}\right)^k \sum_{i=0}^k \beta_i$ from eq. (16).

From the consistency condition

$$\sum_{i=0}^k \beta_i \neq 0$$

Hence if $\beta_k = 0$ and if we require stability as $h \rightarrow \infty$, we find the consistency condition is violated. The conclusion is that no explicit linear multistep method can be stable as $h \rightarrow \infty$.

GEAR'S FORMULA

$$\underline{k = 2} \quad y_{n+1} = \frac{4}{3} y_n - \frac{1}{3} y_{n-1} + h \left(\frac{2}{3} \dot{y}_{n+1} \right)$$

$$\underline{k = 3} \quad y_{n+1} = \frac{18}{11} y_n - \frac{9}{11} y_{n-1} + \frac{2}{11} y_{n-2} + h \left(\frac{6}{11} \dot{y}_{n+1} \right)$$

$$\underline{k = 4} \quad y_{n+1} = \frac{48}{25} y_n - \frac{36}{25} y_{n-1} + \frac{16}{25} y_{n-2} - \frac{3}{25} y_{n-3} + h \left(\frac{12}{25} \dot{y}_{n+1} \right)$$

$$\underline{k = 5} \quad y_{n+1} = \frac{300}{137} y_n - \frac{300}{137} y_{n-1} + \frac{200}{137} y_{n-2} - \frac{75}{137} y_{n-3} + \frac{12}{137} y_{n-4} + h \left(\frac{60}{137} \dot{y}_{n+1} \right)$$

$$\underline{k = 6} \quad y_{n+1} = \frac{360}{147} y_n - \frac{450}{147} y_{n-1} + \frac{400}{147} y_{n-2} - \frac{225}{147} y_{n-3} + \frac{72}{147} y_{n-4} - \frac{10}{147} y_{n-5} + h \left(\frac{60}{147} \dot{y}_{n+1} \right)$$

Coefficients of Optimized Formula (k = 4):

$\alpha_0 = 1.584$	$\beta_{-1} = 0.4539$
$\alpha_1 = -1.017$	$\beta_0 = 0.235$
$\alpha_2 = 0.529$	$\beta_1 = 0.0568$
$\alpha_3 = -0.0968$	$\beta_2 = 0.00567$
	$\beta_3 = 0.000201$

TRUNCATION ERROR

An error term for a Taylor series expansion of $y(t)$ about a point in time at nh can be obtained from the integral

$$\frac{1}{r!} \int_{nh}^{(n-i)h} ([n-i]h-s)^r y^{(r+1)}(s) ds$$

If this is integrated by parts, we obtain

$$\begin{aligned} & \frac{1}{r!} \int_{nh}^{(n-i)h} [n-i]h-s)^r y^{(r+1)}(s) ds = \frac{1}{r!} ([n-i]h-s)^r y^{(r)}(s) \Big|_{nh}^{(n-i)h} \\ & + \frac{1}{(r-1)!} \int_{nh}^{(n-i)h} ([n-i]h-s)^{r-1} y^{(r)}(s) ds \\ & = \frac{-(-ih)^r}{r!} y^{(r)}(nh) + \frac{1}{(r-1)!} \int_{nh}^{(n-i)h} ([n-i]h-s)^{r-1} y^{(r)}(s) ds \end{aligned}$$

If integration by parts is carried out r times and the terms rearranged:

$$\begin{aligned} y([n-i]h) &= y(nh) - ih y^{(1)}(nh) + \frac{i^2 h^2}{2!} y^{(2)}(nh) - \dots \\ &+ \frac{(-1)^r i^r h^r}{r!} y^{(r)}(nh) + \frac{1}{r!} \int_{nh}^{(n-i)h} ([n-i]h-s)^r y^{(r+1)}(s) ds \quad (1) \end{aligned}$$

A Taylor series expansion with error term can be obtained in the same manner for the first derivative of $y(t)$ about a point nh .

$$\begin{aligned} y^{(1)}([n-i]h) &= y^{(1)}(nh) - ih y^{(2)}(nh) + \dots \\ &+ \frac{(-1)^{r-1} i^{r-1} h^{r-1}}{(r-1)!} y^{(r)}(nh) + \frac{1}{(r-1)!} \int_{nh}^{(n-i)h} ([n-i]h-s)^{r-1} y^{(r+1)}(s) ds \quad (2) \end{aligned}$$

Equations (1) and (2) can be substituted into the integration formula in eq. (3) and an expression for T_n can be obtained.

$$y_{n+1} = \sum_{i=0}^{k-1} \alpha_i y_{n-i} + h \sum_{i=-1}^{k-1} \beta_i \dot{y}_{n-i} + T_n \quad (3)$$

Since eq. (3) is exact up to and including order p we obtain for T_n :

$$T_n = \frac{1}{p!} \int_{(n+1-k)h}^{(n+1)h} \left\{ \overline{([n+1]h-s)}^p - ph\beta_{-1} \overline{([n+1]h-s)}^{p-1} \right. \\ \left. + \sum_{i=1}^{k-1} \left[\alpha_i \overline{([n-i]h-s)}^p + ph\beta_i \overline{([n-i]h-s)}^{p-1} \right] \right\} y^{(p+1)}(s) ds \quad (4)$$

where

$$\overline{([n-i]h-s)} = \begin{cases} [n-i]h-s & \begin{cases} [n-i]h \leq s \leq nh, i \neq -1 \\ nh \leq s, i = -1 \end{cases} \\ 0 & \text{otherwise} \end{cases}$$

Equation (4) can be written

$$T_n = \frac{1}{p!} \int_{(n+1-k)h}^{(n+1)h} G(s) y^{(p+1)}(s) ds \quad (5)$$

If $G(s)$ is of the same sign over the interval $[(n+1-k)h, (n+1)h]$, then $\int G(s) ds$ is a monotonic function. If $y^{(p+1)}(s)$ is continuous over the interval, then the First Mean Value Theorem applies.¹² The estimate for the truncation error can be written:

$$T_n = \frac{1}{p!} y^{(p+1)}(\eta) \int_{(n+1-k)h}^{(n+1)h} G(s) ds \quad (6)$$

where $\eta \in [(n+1-k)h, (n+1)h]$.

Example

Let $k = p = 4$

$$y_{n+1} = \alpha_0 y_n + \alpha_1 y_{n-1} + \alpha_2 y_{n-2} + \alpha_3 y_{n-3} \\ + h \left\{ \beta_{-1} \dot{y}_{n+1} + \beta_0 \dot{y}_n + \beta_1 \dot{y}_{n-1} + \beta_2 \dot{y}_{n-2} + \beta_3 \dot{y}_{n-3} \right\} + T_n \quad (7)$$

The influence function is obtained from eq. (4).

$$G(s) = G_1(s) = \alpha_3 ([n-3]h-s)^4 + 4h\beta_3 ([n-3]h-s)$$

$$\text{for } [n-3]h \leq s \leq [n-2]h$$

$$G(s) = G_2(s) = G_1(s) + \alpha_2 ([n-2]h-s)^4 + 4h\beta_2 ([n-2]h-s)^3$$

$$\text{for } [n-2]h \leq s \leq [n-1]h$$

$$G(s) = G_3(s) = G_2(s) + \alpha_1 ([n-1]h-s)^4 + 4h\beta_1 ([n-1]h-s)^3$$

$$\text{for } [n-1]h \leq s \leq nh$$

$$G(s) = ([n+1]h-s)^4 - 4h\beta_{-1} ([n+1]h-s)^3$$

$$\text{for } nh \leq s \leq [n+1]h$$

For Gear's Method

$$\alpha_0 = \frac{48}{25}, \quad \alpha_1 = -\frac{36}{25}, \quad \alpha_2 = \frac{16}{25}, \quad \alpha_3 = -\frac{3}{25}, \quad \beta_{-1} = \frac{12}{25},$$

$$\beta_0 = \beta_1 = \beta_2 = \beta_3 = 0$$

The influence function is shown in Figure 1. Since the influence function is of the same sign over the interval $([n-3]h, [n+1]h)$, the error is

$$T_n = \frac{E_4}{5!} \cdot y^{(5)}(\eta) h^5$$

To find E_4 assume $y(t) = t^5$. Substituting this into the integration formula:

$$h^5 = -\alpha_1 h^5 - 32\alpha_2 h^5 - 243\alpha_3 h^5 + 5h^5 (\beta_{-1} + \beta_1 + 16\beta_2 + 81\beta_3) + E_4 h^5$$

Solving for E_4 :

$$E_4 = 1 + \alpha_1 + 32\alpha_2 + 243\alpha_3 - 5(\beta_{-1} + \beta_1 + 16\beta_2 + 81\beta_3)$$

Using the values of the α 's and β 's for Gear's Method gives

$$E_4 = -11.52 \left| T_n \right| = \frac{2.304}{4!} h^5 \left| y^{(5)}(\eta) \right|$$

This shows that

$$\left| \int_{(n-3)h}^{(n+1)h} G(s) ds \right| = 2.304h^5$$

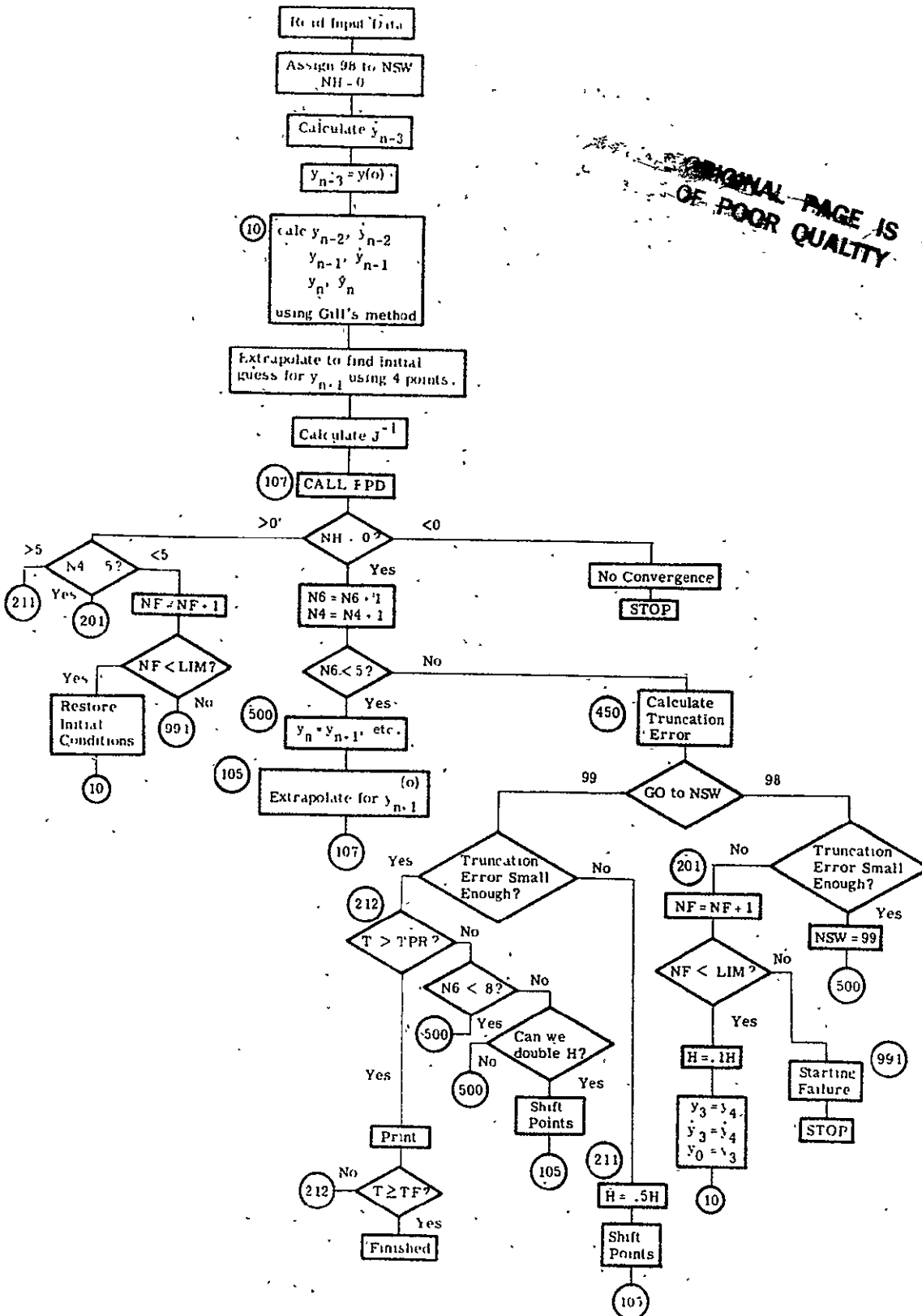
which can be verified by integrating the influence function directly.

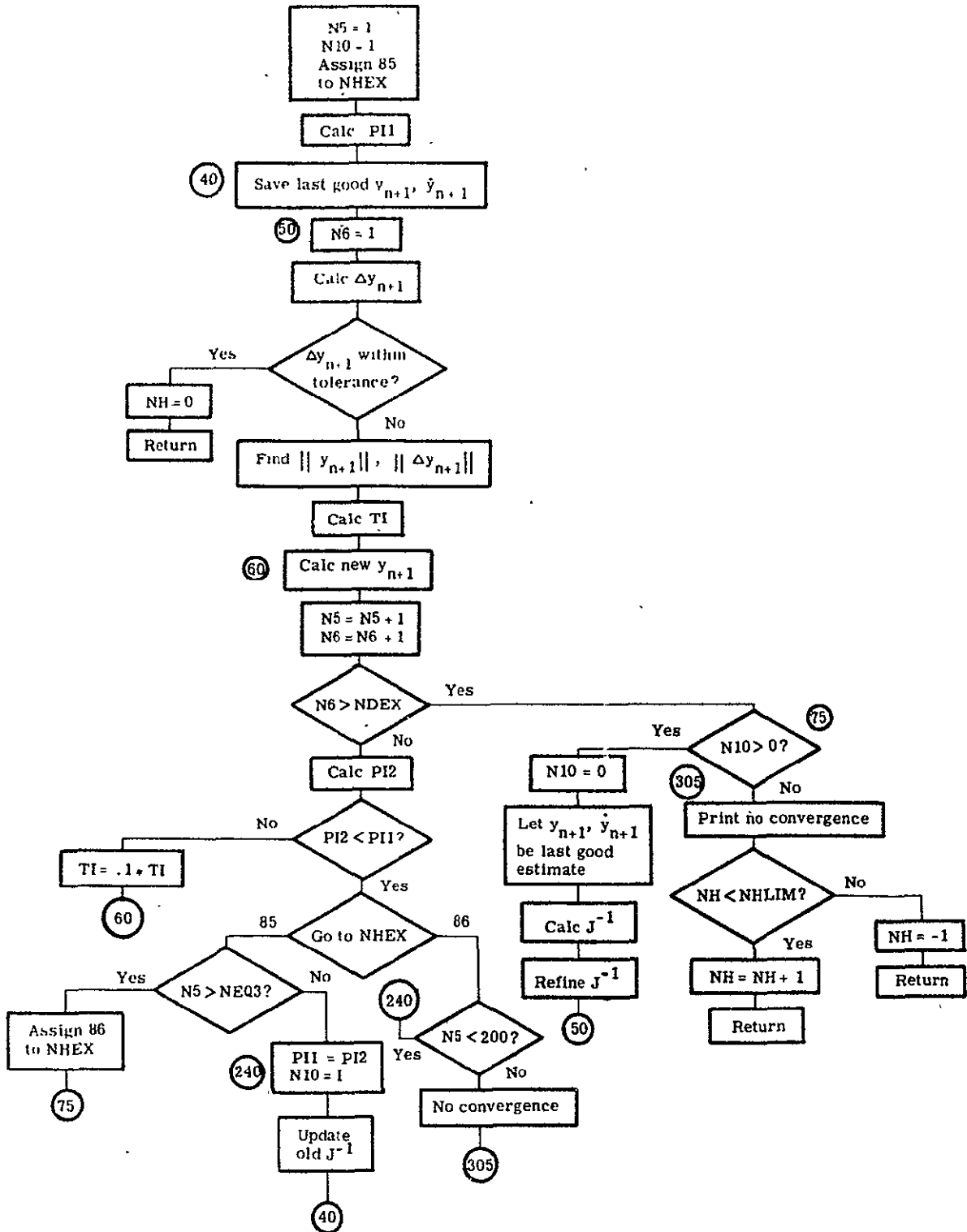
APPENDIX B

SOURCE LISTING

FLOW CHART OF PROGRAM

SAMPLE PROBLEM





```

1*      CMAIN
2*      COMMON YB,DY0,DB,T,H,NV,NUM,Y,Y1,Y2,Y3,Y4,E1,E2,DY,DY1,DY2,DY3,
3*      1 DY4,TPR,DT,NDEX,JAC,Q,G,N5
4*      COMMON/BND5/YLIM,XNLIM,ELIM,SC,NEQ3,NH
5*      DOUBLE PRECISION YO(10),DY0(10),DB(10),Y(10),Y1(10),Y2(10),Y3(10)
6*      1 Y4(10),DY(10),DY1(10),DY2(10),DY3(10),DY4(10),Q(10),G(10,10),
7*      2 Y5(10),Y6(10),Y7(10),DY5(10),DY6(10),DY7(10)
8*      DOUBLE PRECISION X,QMAX,SET
9*      DATA LIM/9/,HMN/1.0E-15/
10*     C
11*     C      INITIALIZE = NF COUNTS STARTING FAILURES
12*     C      N4 COUNTS SUCCESSFUL INTEGRATIONS
13*     C      N6 COUNTS STEPS SINCE LAST HALVING OR DOUBLING
14*     C      NH FLAGS CONVERGENCE IN FPD
15*     C      LIM IS THE STARTING FAILURE LIMIT
16*     C      JAC COUNTS MATRIX INVERSIONS = JACOBIAN EVALUATION
17*     C      NUM COUNTS EVALUATIONS OF STATE EQUATIONS
18*     PRINT 3
19*     3 FORMAT(21H1 INITIAL CONDITIONS)
20*     ASSIGN 98 TO NSW
21*     NH = 0
22*     NF = 0
23*     JAC = 0
24*     NUM = 0
25*     READ 5,TF,PT,NV
26*     5 FORMAT(2F8.1,15)
27*     NEQ3 = MAX0(NV*3,7)
28*     NDEX = 6
29*     DT = TF/PT
30*     TPR = DT
31*     H = TF * 0.001
32*     READ 6,(Y0(I),I=1,NV)
33*     6 FORMAT(10F8,2)
34*     PRINT 4,(Y0(I),I=1,NV)
35*     4 FORMAT(1X,8E15.7)
36*     READ 101,YLIM,ELIM,XNLIM,SC
37*     101 FORMAT(4E7.1)
38*     PRINT 100
39*     100 FORMAT(9H0 LIMITS)
40*     PRINT 4, YLIM,ELIM,XNLIM,SC
41*     ELIM = ELIM * SC
42*     EDUB = ELIM/40.
43*     CALL STATE (Y0,DY3)
44*     DO 261 I=1,NV
45*     261 Y3(I) = Y0(I)
46*     C
47*     C      USE GILL'S METHOD TO START
48*     10 T = 0.
49*     N4 = 3
50*     N6 = 3
51*     CALL GILL
52*     DO 262 I=1,NV
53*     DY2(I) = DY0(I)
54*     262 Y2(I) = Y0(I)
55*     CALL GILL
56*     DO 263 I=1,NV
57*     DY1(I) = DY0(I)
58*     263 Y1(I) = Y0(I)
59*     CALL GILL
60*     C

```

```

61*      C      INITIALLY, THE 4 POINT SCHEME IS USED
62*      DO 110 I=1,NV
63*      110 Y(I) = 4.*(Y0(I) + Y2(I))-6.*Y1(I)-Y3(I)
64*      T = T+H
65*      C
66*      C      CALCULATE INITIAL JACOBIAN
67*      CALL STATE (Y,DY)
68*      CALL QX(DB,QMAX)
69*      DO 13 J=1,NV
70*      X = Y(J)
71*      DIF = DABS(X) - YLIM
72*      IF (DIF) 16,16,15
73*      16 Y(I) = X + 1.0D-4
74*      SET = 1.0D4
75*      GO TO 18
76*      15 Y(I) = X*1.0D10
77*      SET = 1000./X
78*      18 CALL STATE (Y,DY)
79*      CALL QX(Q,QMAX)
80*      DO 14 J=1,NV
81*      14 G(J,I) = (Q(J)-DB(J))*SET
82*      13 Y(I) = X
83*      CALL MATINV (NV,G)
84*      JAC = JAC + 1
85*      GO TO 107
86*      C
87*      C      TO HERE WHEN H IS TO BE DOUBLED
88*      222 H = H*H
89*      N6 = 4
90*      C
91*      C      SHIFT Y*5 = Y1 IS ALREADY IN POSITION
92*      DO 180 I=1,NV
93*      DY0(I) = DY(I)
94*      DY2(I) = DY3(I)
95*      DY3(I) = DY5(I)
96*      DY4(I) = DY7(I)
97*      Y0(I) = Y(I)
98*      Y2(I) = Y3(I)
99*      Y3(I) = Y5(I)
100*      180 Y4(I) = Y7(I)
101*      C
102*      C      USE 5 POINT FORMULA ONCE STARTED
103*      105 DO 111 I=1,NV
104*      111 Y(I) = 5.*(Y0(I) + 2.*(Y2(I)-Y1(I)) - Y3(I)) + Y4(I)
105*      T = T+H
106*      107 CALL FPD
107*      IF (NH) 305,449,611
108*      611 IF (N4=4) 951,201,211
109*      449 N6 = N6+1
110*      N4 = N4+1
111*      PRINT 9,H,N5,N6,N4
112*      9 FORMAT(3HH =,E12.5,5H N5 =,I4,5H N6 =,I5,5H N4 =,I5)
113*      IF (N6=5) 500,450,450
114*      202 ASSIGN 99 TO NSW
115*      C
116*      C      GOOD POINT - DOWNSHIFT AND MOVE ON
117*      500 DO 120 I=1,NV
118*      Y7(I) = Y6(I)

```



```

119*      Y6(I) = Y5(I)
120*      Y5(I) = Y4(I)
121*      Y4(I) = Y3(I)
122*      Y3(I) = Y2(I)
123*      Y2(I) = Y1(I)
124*      Y1(I) = Y0(I)
125*      Y0(I) = Y(I)
126*      DY7(I) = DY6(I)
127*      DY6(I) = DY5(I)
128*      DY5(I) = DY4(I)
129*      DY4(I) = DY3(I)
130*      DY3(I) = DY2(I)
131*      DY2(I) = DY1(I)
132*      DY1(I) = DY0(I)
133*      120 DY0(I) = DY(I)
134*      GO TO 105
135*      C
136*      C ----- NEED 5 PAST POINTS PLUS PRESENT POINT FOR FIFTH DERIVATIVE -----
137*      450 CALL D5Y
138*      C
139*      C      D5Y RETURNS E1 = MAX(ABS(YV)*DLIM-ABS(Y)*ELIM)
140*      C      AND E2 = MAX(ABS(YV)*DLIM-ABS(Y)*ELIM/40.)
141*      GO TO NSW,(98,99)
142*      C
143*      C      THIS SEQUENCE ONLY PERFORMED IN STARTING
144*      98 IF (E1 - ELIM) 202,202,201
145*      C
146*      C      ALLOW LIM SUCCESSIVE STARTING FAILURES
147*      951 NF = NF+1
148*      IF (NF-LIM) 952,991,991
149*      201 NF = NF+1
150*      IF (NF-LIM) 950,991,991
151*      C
152*      C      REDUCE STEP AND RESTORE INITIAL CONDITIONS FOR N4 = 3
153*      952 H = H*0.1
154*      DO 953 I=1,NV
155*      953 Y0(I) = Y3(I)
156*      GO TO 10
157*      C      REDUCE STEP AND RESTORE INITIAL CONDITIONS FOR N4 = 4
158*      C
159*      950 H = H*0.1
160*      DO 160 I=1,NV
161*      DY3(I) = DY4(I)
162*      Y3(I) = Y4(I)
163*      160 Y0(I) = Y4(I)
164*      GO TO 10
165*      99 IF (E1 - ELIM) 212,212,211
166*      C
167*      C      ERROR OUT OF BOUNDS - HALVE STEP SIZE AND TRY AGAIN
168*      211 H = H*0.5
169*      IF (H-HMIN) 990,940,940
170*      940 T = T - 5.0*H
171*      DO 130 I=1,NV
172*      QMAX = Y1(I)
173*      SET = Y3(I)
174*      Y1(I) = (35.*Y0(I)+140.*QMAX+70.*Y2(I)+28.*SET-5.*Y4(I))*0.0078125
175*      Y3(I) = (-5.*Y0(I)+60.*QMAX+90.*Y2(I)-20.*SET+3.*Y4(I))*0.0078125
176*      Y4(I) = Y2(I)

```

```

177*      DY4(I) = DY2(I)
178*      DY2(I) = DY1(I)
179*      130 Y2(I)=QMAX
180*      N6 = -4
181*      CALL STATE (Y3,DY3)
182*      T = T+H+H
183*      CALL STATE (Y1,DY1)
184*      T = T+H
185*      GO TO 105
186*      C
187*      C      ERROR IN BOUNDS - CHECK WHETHER TO PRINT
188*      212 IF (T-TPR) 302,301,301
189*      301 CALL PRYNT
190*      IF (TPR-TF) 212,212,304
191*      304 IF (T-TF) 302,306,306
192*      C
193*      C      CHECK WHETHER TO DOUBLE STEP SIZE - DELAY OF 3 PASSES
194*      C      IS MANDATED BY STRUCTURE OF POINT-SAVING MECHANISM
195*      302 IF (N6-8) 500,480,480
196*      480 IF (E2-EDUB) 222,222,500
197*      991 PRINT 1,NF,E1
198*      1 FORMAT(1H0,12,28H STARTING FAILURES = ERROR =,E15.7)
199*      990 PRINT 2,H,T
200*      2 EFORMAT(21H0STEP TOO SMALL = H =,E15.7,4H T =,E15.7)
201*      305 PRINT 78,(Y(I),YD(I),Y1(I),YZ(I),Y3(I),Y4(I),Y5(I),Y6(I),I=1,NV)
202*      78 FORMAT(1H0,6E17.8)
203*      306 PRINT 7,NUM
204*      7 FORMAT(12H0STATE EQUATIONS EVALUATED,15,6H TIMES)
205*      PRINT 8,JAC
206*      8 FORMAT(19H0JACOBIAN EVALUATED,15,6H TIMES)
207*      STOP
208*      END

```

ND OF COMPILATION: NO DIAGNOSTICS.

```

1*      CFPO
2*      SUBROUTINE FPD
3*      COMMON YO,DYO,DB,T,H,NV,NUM,Y,Y1,Y2,Y3,Y4,E1,E2,DY,DY1,DY2,DY3,
4*      DY4,TPR,DT,NDEX,JAC,Q,G,N5
5*      COMMON/BNDS/YLIM,XNLIM,ELIM,SC,NEQ3,NH
6*      DOUBLE PRECISION YO(10),DYO(10),DB(10),Y(10),Y1(10),Y2(10),Y3(10),
7*      Y4(10),DY(10),DY1(10),DY2(10),DY3(10),DY4(10),Q(10),G(10,10),
8*      EXTRA(10),SAVE(10,10),DQ(10),RES(10,10),OLDB(10),CURDY(10),
9*      B(10)
10*     DOUBLE PRECISION X,QMAX,SET,SUM,DIV
11*     EQUIVALENCE (NV,NV2),(Y,B),(BLIM,YLIM),(OLDB,EXTRA)
12*     DATA LIM/6/,NHLIM/17/
13*     C
14*     C      N5 COUNTS EVALUATIONS OF B
15*     C      N10 FLAGS TWO SUCCESSIVE NDEX STEP FAILURES
16*     C      N6 COUNTS THESE NDEX STEPS
17*     N5=1
18*     N10=1
19*     ASSIGN B5 TO NHEX
20*     C
21*     C      FIND SQRT OF PERFORMANCE INDEX, TO AVOID OVERFLOW
22*     CALL STATE (Y,DY)
23*     CALL QX(DQ,QMAX)
24*     P11=0.
25*     DO 20 I=1,NV2
26*     P11=P11+(DQ(I)/QMAX)**2
27*     P11 = SQRT(P11)*QMAX
28*     DO 163 I=1,NV
29*     CURDY(I) = DY(I)
30*     OLDB(I) = B(I)
31*     N6 = 1
32*     IOUT = NV
33*     DO 25 I=1,NV2
34*     SUM=0.00
35*     DO 35 J=1,NV2
36*     SUM=SUM + G(I,J)*DQ(J)
37*     IF(DABS(SUM)-XNLIM*(SC+ABS(OLDB(I))))199,199,25
38*     IOUT = IOUT-1
39*     25 DB(I)=SUM
40*     C
41*     C      IF ALL IN BOUNDS, PRESENT B'S ARE THE SOLUTION
42*     IF(IOUT)210,210,67
43*     67 BN = 0.
44*     DBN = 0.
45*     DO 68 I=1,NV
46*     BN = BN + B(I)*B(I)
47*     68 DBN = DBN + DB(I)*DB(I)
48*     IF (DBN-BN)63,63,64
49*     63 TI = 1.
50*     GO TO 60
51*     64 TI = SQRT(BN/DBN)
52*     GO TO 60
53*     210 DO 205 I=1,NV
54*     DY(I) = CURDY(I)
55*     NH = 0
56*     RETURN
57*     C
58*     C      IF NORM OF DB TOO LARGE, RECALCULATE B AND GET NEW P I
59*     90 TI=,1*TI
60*     DO 55 I=1,NV2

```

```

61*      55 B(I)=OLDB(I) - T1*DB(I)
62*      N5=N5+1
63*      N6=N6+1
64*      C
65*      C      IF NDEX FAILURES, CALCULATE NEW G - HALVE IF THIS FAILS
66*      IF (N6=NDEX) 70,70,75
67*      75 IF (N10) 230,230,220
68*      230 PRINT 77,N5,H,T
69*      77 FORMAT(26HNO CONVERGENCE,N10=0, N5=,I4,4H H =,E15.7,4H T =,E15.7)
70*      305 PRINT 78,(Y(I),Y0(I),Y1(I),Y2(I),Y3(I),Y4(I),I=1,NV)
71*      78 FORMAT(1X,6 E16.8)
72*      IF (NH-NHLIM) 575,576,576
73*      575 NH = NH+1
74*      RETURN
75*      576 NH = -1
76*      RETURN
77*      220 N10=0
78*      C
79*      C      MUST RESTORE OLD B'S
80*      DO 161 I=1,NV
81*      161 B(I) = OLDB(I)
82*      C
83*      C      FIND G
84*      DO 13 I=1,NV2
85*      X = B(I)
86*      DIF = DABS(X) - BLIM
87*      IF (DIF) 16,16,15
88*      16 B(I) = X + 1.0D-4
89*      SET = 1.0D4
90*      GO TO 18
91*      15 B(I) = X*1.0D100
92*      SET = 1000./X
93*      18 CALL STATE (Y,DY)
94*      CALL QX(Q,QMAX)
95*      DO 14 J=1,NV2
96*      G(J,I)=(Q(J)-DQ(J))*SET
97*      14 SAVE(J,I) = G(J,I)
98*      13 B(I)=X
99*      CALL MATINV(NV2,G)
100*      JAC=JAC+1
101*      C
102*      C      ITERATE ON THE RESIDUAL
103*      DO 302 M=1,LIM
104*      DO 415 I=1,NV
105*      DO 412 J=1,NV
106*      SUM = 0.0D0
107*      DO 413 K=1,NV
108*      413 SUM = SUM + SAVE(I,K)*G(K,J)
109*      412 RES(I,J) = -SUM
110*      415 RES(I,I) = 2.0 + RES(I,I)
111*      DO 420 I=1,NV
112*      DO 417 J=1,NV
113*      SUM = 0.0D0
114*      DO 418 K=1,NV
115*      418 SUM = SUM + G(I,K)*RES(K,J)
116*      417 DB(J) = SUM
117*      DO 419 J=1,NV
118*      419 G(I,J) = DB(J)

```

```

119*      420 CONTINUE
120*      302 CONTINUE
121*      GO TO 50
122*      C
123*      C      TO HERE IF LESS THAN NDEX FAILURES
124*      70 CALL STATE (Y,DY)
125*      CALL QX(Q,QMAX)
126*      P12=0.
127*      DO 80 I=1,NV2
128*      80 P12=P12+(Q(I)/QMAX)**2
129*      P12 = SQRT(P12)*QMAX
130*      C
131*      C      CHECK NEW P 1
132*      IF(P12-P11) 87,90,90
133*      87 GO TO NHEX,(85,86)
134*      86 IF(N5=200) 240,250,250
135*      250 PRINT 260,H,T
136*      260 FORMAT(27HONO CONVERGENCE,N5=200, H =,E15.7,4H T =,E15.7)
137*      GO TO 305
138*      85 IF (N5=NEQ3) 240,240,245
139*      245 ASSIGN 86 TO NHEX
140*      GO TO 75
141*      C
142*      C      ITERATE TO GET NEW G AND TRY AGAIN
143*      240 P11=P12
144*      N10=1
145*      DO 110 I=1,NV2
146*      110 DQ(I)=Q(I)-DQ(I)
147*      DIV=0.00
148*      DO 150 I=1,NV2
149*      SUM=0.00
150*      DO 151 J=1,NV2
151*      151 SUM=SUM+DQ(J)*G(I,J)
152*      DIV=DIV+DB(I)*SUM
153*      150 EXTRA(I)= SUM + T1 * DB(I)
154*      DIV = DIV
155*      DO 155 I=1,NV2
156*      SUM = EXTRA(I)/DIV
157*      DO 154 J=1,NV2
158*      154 SAVE(I,J)=DB(J)*SUM
159*      155 SAVE(I,I)=1.+SAVE(I,I)
160*      DO 156 K=1,NV2
161*      DO 157 I=1,NV2
162*      SUM=0.00
163*      DO 158 J=1,NV2
164*      158 SUM=SUM+SAVE(I,J)*G(J,K)
165*      157 EXTRA(I)=SUM
166*      DO 159 I=1,NV2
167*      159 G(I,K)=EXTRA(I)
168*      156 DQ(K)=Q(K)
169*      GO TO 40
170*      END

```

END OF COMPILATION: NO DIAGNOSTICS.

```

1* CPRINT
2* SUBROUTINE PRYNT
3* COMMON Y0,DY0,DB,T,H,NV,NUM,Y,Y1,Y2,Y3,Y4,E1,E2,DY,DY1,DY2,DY3,
4* 1 DY4,TPR,DT
5* DOUBLE PRECISION Y0(10),DY0(10),DB(10),Y(10),Y1(10),Y2(10),Y3(10),
6* 1 Y4(10),DY1(10),DY2(10),DY3(10),DY4(10)
7* DIMENSION V(6),Z(10)
8* EQUIVALENCE (Z,DB)
9* C
10* C LAGRANGIAN INTERPOLATION TO MAKE THE TIME COME OUT NICE
11* A1 = T
12* DO 10 J=1,6
13* PROD = 1
14* D = T
15* DO 12 K=1,6
16* IF(K=J) 20,12,20
17* 20 PROD = PROD*(TPR-D)/(A1-D)
18* 12 D = D-H
19* V(J) = PROD
20* 10 A1 = A1-H
21* DO 30 J=1,NV
22* 30 Z(J)=V(1)*Y(J)+V(2)*Y0(J)+V(3)*Y1(J)+V(4)*Y2(J)+V(5)*Y3(J)
23* 1 +V(6)*Y4(J)
24* PRINT 40,TPR
25* 40 FORMAT(4H0T =,E15.7)
26* PRINT 60,(Z(I),I=1,NV)
27* 60 FORMAT (2X,5G15.6)
28* TPR=TPR+DT
29* RETURN
30* END

```

END OF COMPILATION: NO DIAGNOSTICS.

```

1*      CGILL
2*      SUBROUTINE GILL
3*      COMMON YO,DYO,U,T,H,NV,NUM
4*      DOUBLE PRECISION YO(10),DYO(10),U(10),A(4,4)
5*      DATA A/.500,.292893218813452476D0,1.70710678118654752D0,.166666666
6*      1.666666667D0,0.D0,.292893218813452476D0,1.70710678118654752D0,.3333
7*      233333333333333D0,1.D0,.585786437626904951D0,3.41421356237309505D0,
8*      3 2*0.D0,.121320343559642573D0,4.12132034355964257D0,0.D0/
9*      C
10*     C      FOURTH ORDER RUNGE-KUTTA METHOD, RESULT IN YO ARRAY UPON EXIT
11*     CALL STATE(YO,DYO)
12*     DO 500 J=1,NV
13*       500 U(J)=0.D0
14*       J1 = 1
15*       DO 1050 J=1,NV
16*         YO(J) = YO(J) + H*(A(J1,1)*DYO(J) - A(J1,2)*U(J))
17*         1050 U(J) = A(J1,3)*DYO(J) - A(J1,4)*U(J)
18*         GO TO (1060,1010,1060,950),J1
19*         1060 T=T+H*0.5
20*         1010 J1 = J1 + 1
21*         CALL STATE(YO,DYO)
22*         GO TO 1040
23*         950 CALL STATE(YO,DYO)
24*         RETURN
25*         END

```

END OF COMPILATION: NO DIAGNOSTICS.

```

1*      COSY
2*      SUBROUTINE D5Y
3*      COMMON YO,DYO,DB,T,H,NV,NUM,Y,Y1,Y2,Y3,Y4,E1,E2
4*      DOUBLE PRECISION YO(10),DYO(10),DB(10),Y(10),Y1(10),Y2(10),Y3(10),
5*      Y4(10)
6*      COMMON/BNDS/YLIM,XNLIM,ELIM
7*      DATA DLIM/.0805/
8*      C
9*      C      CHOOSE MAXIMUM TRUNCATION ERROR (SORT OF)
10*     E1 = 1.0E20
11*     E2 = E1
12*     DO 20 J=1,NV
13*       ER = ABS(Y(J)+5.*(Y3(J)+2.*(Y1(J)-Y2(J)) - YO(J))-Y4(J)) * DLIM
14*       ES = ABS(Y(J))*ELIM
15*       ET = ER+ES
16*       IF(E1-ET) 30,10,10
17*       30 E1 = ET
18*       10 ET = ER+ES/40.
19*       IF(E2-ET) 40,20,20
20*       40 E2 = ET
21*       20 CONTINUE
22*       RETURN
23*       END

```

END OF COMPILATION: NO DIAGNOSTICS.

```

1*  C:QX
2*  SUBROUTINE QX(Q,Z)
3*  COMMON Y0,DY0,DB,T,H,NV,NUM,Y,Y1,Y2,Y3,Y4,E1,E2,DY,DY1,DY2,DY3
4*  DOUBLE PRECISION Y0(10),DY0(10),DB(10),Y(10),Y1(10),Y2(10),Y3(10),
5*  Y4(10),DY1(10),DY2(10),DY3(10)
6*  DOUBLE PRECISION Q,Z,X
7*  DIMENSION Q(1)
8*  DOUBLE PRECISION BM,B0,B2,B3,B1,A0,A1,A2,A3
9*  DATA A0/1.584299999999998677D0/
10*  DATA A1/-1.016965299999996671D0/
11*  DATA A2/.5294440499999970988D0/
12*  DATA A3/-.9677874999999910463D-1/
13*  DATA BM/.453866922117471398D0/
14*  DATA B0/.235026309446787243D0/
15*  DATA B1/-.568199764548261324D-1/
16*  DATA B2/.567200528010787405D-2/
17*  DATA B3/-.201336700809476272D-3/
18*  C
19*  C   TAKE DIFFERENCE OF Y PREDICTED (ORIGINALLY FROM EXTRAPOLATION)
20*  C   AND Y CALCULATED (BASED ON INTEGRATION FORMULA)
21*  Z = 0.00
22*  DO 30 J=1,NV
23*  Q(J) = Y(J) - (Y0(J)*A0 + Y1(J)*A1 + Y2(J)*A2 + Y3(J)*A3 +
24*  1 H*(DY(J)*BM + DY0(J)*B0 + DY1(J)*B1 + DY2(J)*B2 + DY3(J)*B3))
25*  X = DABS(Q(J))
26*  IF (X=Z) 30,30,40
27*  40 Z = X
28*  30 CONTINUE
29*  RETURN

```

```

1*  CMATINV
2*  SUBROUTINE MATINV(N,Z)
3*  DOUBLE PRECISION Z,X,CC
4*  DIMENSION IP(10),Z(10,1),IX(10,2)
5*  DO 10 I=1,N
6*  10 IP(I)=0
7*  DO 76 I=1,N
8*  X = 0.00
9*  DO 69 J=1,N
10*  IF (IP(J)=1) 61,69,61
11*  61 DO 68 K=1,N
12*  IF (IP(K)=1) 64,68,399
13*  64 CC=DABS(Z(J,K))
14*  IF (X=CC) 67,68,68
15*  67 I2=J
16*  I1=K
17*  X = CC
18*  68 CONTINUE
19*  69 CONTINUE
20*  IP(I1)=IP(I1)+1
21*  IF (I2=I1) 70,72,70
22*  70 DO 71 L=1,N
23*  X=Z(I2,L)
24*  Z(I2,L)=Z(I1,L)
25*  71 Z(I1,L)=X
26*  72 IX(I,1)=I2
27*  IX(I,2)=I1
28*  X = Z(I1,I1)
29*  Z(I1,I1)=1.0
30*  DO 73 L=1,N

```



```

31*      73 Z(I1,L)=Z(I1,L)/X
32*      DO 76 J=1,N
33*      IF(J=I1) 74,76,74
34*      74 X=Z(J,I1)
35*      Z(J,I1)=0.
36*      DO 75 L=1,N
37*      75 Z(J,L)=Z(J,L)-Z(I1,L)*X
38*      76 CONTINUE
39*      L = N+1
40*      DO 79 I=1,N
41*      L = L-1
42*      IF(I-X(L,1)-IX(L,2)) 77,79,77
43*      77 I2=IX(L,1)
44*      I1=IX(L,2)
45*      DO 78 K=1,N
46*      X=Z(K,I2)
47*      Z(K,I2)=Z(K,I1)
48*      Z(K,I1)=X
49*      78 CONTINUE
50*      79 CONTINUE
51*      399 RETURN
52*      END

```

END OF COMPILATION: NO DIAGNOSTICS.

```
1*      CSTATE
2*      SUBROUTINE STATE(X,D)
3*      COMMON YO,DYO,DB,T,H,NV,NUM
4*      DOUBLE PRECISION YO(10),DYO(10),DB(10)
5*      DOUBLE PRECISION X(1),D(1)
6*      {D(2)=X(1)-X(2)}
7*      {D(1)=1000.*(X(1)+D(2))}
8*      NUM=NUM+1
9*      RETURN
10*     END
```

END OF COMPILATION: NO DIAGNOSTICS.

0.1000000E 01 0.1000000E 01

LIMITS

	0.1000000E-05	0.1000000E-03	0.1000000E-04	0.1000000E 01
H = 0.100000E-01 N5 = 26 N6 = 4 N4 = 4				
H = 0.100000E-01 N5 = 2 N6 = 5 N4 = 5				
H = 0.100000E-02 N5 = 2 N6 = 4 N4 = 4				
H = 0.100000E-02 N5 = 2 N6 = 5 N4 = 5				
H = 0.100000E-03 N5 = 2 N6 = 4 N4 = 4				
H = 0.100000E-03 N5 = 2 N6 = 5 N4 = 5				
H = 0.100000E-03 N5 = 2 N6 = 6 N4 = 6				
H = 0.100000E-03 N5 = 2 N6 = 7 N4 = 7				
H = 0.100000E-03 N5 = 2 N6 = 8 N4 = 8				
H = 0.100000E-03 N5 = 2 N6 = 9 N4 = 9				
H = 0.200000E-03 N5 = 3 N6 = 5 N4 = 10				
H = 0.200000E-03 N5 = 2 N6 = 6 N4 = 11				
H = 0.200000E-03 N5 = 2 N6 = 7 N4 = 12				
H = 0.200000E-03 N5 = 2 N6 = 8 N4 = 13				
H = 0.200000E-03 N5 = 2 N6 = 9 N4 = 14				
H = 0.200000E-03 N5 = 2 N6 = 10 N4 = 15				
H = 0.200000E-03 N5 = 2 N6 = 11 N4 = 16				
H = 0.200000E-03 N5 = 2 N6 = 12 N4 = 17				
H = 0.200000E-03 N5 = 2 N6 = 13 N4 = 18				
H = 0.200000E-03 N5 = 2 N6 = 14 N4 = 19				
H = 0.400000E-03 N5 = 3 N6 = 5 N4 = 20				
H = 0.200000E-03 N5 = 3 N6 = 5 N4 = 21				
H = 0.200000E-03 N5 = 2 N6 = 6 N4 = 22				
H = 0.200000E-03 N5 = 2 N6 = 7 N4 = 23				
H = 0.200000E-03 N5 = 2 N6 = 8 N4 = 24				
H = 0.200000E-03 N5 = 2 N6 = 9 N4 = 25				
H = 0.200000E-03 N5 = 2 N6 = 10 N4 = 26				
H = 0.400000E-03 N5 = 3 N6 = 5 N4 = 27				
H = 0.400000E-03 N5 = 2 N6 = 6 N4 = 28				
H = 0.400000E-03 N5 = 2 N6 = 7 N4 = 29				
H = 0.400000E-03 N5 = 2 N6 = 8 N4 = 30				
H = 0.400000E-03 N5 = 2 N6 = 9 N4 = 31				
H = 0.800000E-03 N5 = 3 N6 = 5 N4 = 32				
H = 0.800000E-03 N5 = 2 N6 = 6 N4 = 33				
H = 0.800000E-03 N5 = 1 N6 = 7 N4 = 34				
H = 0.800000E-03 N5 = 2 N6 = 8 N4 = 35				
H = 0.800000E-03 N5 = 2 N6 = 9 N4 = 36				
H = 0.160000E-02 N5 = 3 N6 = 5 N4 = 37				
H = 0.160000E-02 N5 = 2 N6 = 6 N4 = 38				
H = 0.160000E-02 N5 = 1 N6 = 7 N4 = 39				
H = 0.160000E-02 N5 = 1 N6 = 8 N4 = 40				
H = 0.320000E-02 N5 = 3 N6 = 5 N4 = 41				
H = 0.320000E-02 N5 = 1 N6 = 6 N4 = 42				
H = 0.320000E-02 N5 = 1 N6 = 7 N4 = 43				
H = 0.320000E-02 N5 = 2 N6 = 8 N4 = 44				
H = 0.640000E-02 N5 = 3 N6 = 5 N4 = 45				
H = 0.640000E-02 N5 = 1 N6 = 6 N4 = 46				
H = 0.640000E-02 N5 = 2 N6 = 7 N4 = 47				
H = 0.640000E-02 N5 = 2 N6 = 8 N4 = 48				
H = 0.640000E-02 N5 = 2 N6 = 9 N4 = 49				
H = 0.640000E-02 N5 = 2 N6 = 10 N4 = 50				
H = 0.128000E-01 N5 = 3 N6 = 5 N4 = 51				
H = 0.128000E-01 N5 = 2 N6 = 6 N4 = 52				
H = 0.128000E-01 N5 = 2 N6 = 7 N4 = 53				
H = 0.128000E-01 N5 = 1 N6 = 8 N4 = 54				
H = 0.256000E-01 N5 = 2 N6 = 5 N4 = 55				
H = 0.256000E-01 N5 = 1 N6 = 6 N4 = 56				
H = 0.256000E-01 N5 = 2 N6 = 7 N4 = 57				
H = 0.256000E-01 N5 = 2 N6 = 8 N4 = 58				

ORIGINAL PAGE IS
OF POOR QUALITY

X = 0.25400E-01	N5 =	2 N6 =	9 N4 =	59
X = 0.25400E-01	N5 =	2 N6 =	10 N4 =	60
X = 0.25400E-01	N5 =	1 N6 =	11 N4 =	61
X = 0.51200E-01	N5 =	2 N6 =	5 N4 =	62
X = 0.51200E-01	N5 =	2 N6 =	6 N4 =	63
X = 0.51200E-01	N5 =	2 N6 =	7 N4 =	64
X = 0.51200E-01	N5 =	2 N6 =	8 N4 =	65

T = 0.5000000E 00				
0.359622 0.779044				
X = 0.51200E-01	N5 =	2 N6 =	9 N4 =	66
X = 0.51200E-01	N5 =	1 N6 =	10 N4 =	67
X = 0.10240E 00	N5 =	3 N6 =	5 N4 =	68
X = 0.10240E 00	N5 =	1 N6 =	6 N4 =	69
X = 0.10240E 00	N5 =	1 N6 =	7 N4 =	70
X = 0.10240E 00	N5 =	2 N6 =	8 N4 =	71

T = 0.1000000E 01				
0.303454 0.606760				
X = 0.20480E 00	N5 =	3 N6 =	5 N4 =	72
X = 0.20480E 00	N5 =	2 N6 =	6 N4 =	73
X = 0.20480E 00	N5 =	2 N6 =	7 N4 =	74

T = 0.1500000E 01				
0.236346 0.472573				
X = 0.20480E 00	N5 =	2 N6 =	8 N4 =	75
X = 0.20480E 00	N5 =	3 N6 =	9 N4 =	76

T = 0.2000000E 01				
0.184070 0.368061				
X = 0.20480E 00	N5 =	2 N6 =	10 N4 =	77
X = 0.20480E 00	N5 =	2 N6 =	11 N4 =	78
X = 0.20480E 00	N5 =	2 N6 =	12 N4 =	79

T = 0.2500000E 01				
0.143362 0.286662				
H = 0.40960E 00	N5 =	4 N6 =	5 N4 =	80

T = 0.3000000E 01				
0.111552 0.223256				
H = 0.40960E 00	N5 =	2 N6 =	6 N4 =	81
H = 0.40960E 00	N5 =	2 N6 =	7 N4 =	82

T = 0.3500000E 01				
0.869545E-01 0.173853				
H = 0.40960E 00	N5 =	2 N6 =	8 N4 =	83

T = 0.4000000E 01				
0.677053E-01 0.135392				
H = 0.40960E 00	N5 =	2 N6 =	9 N4 =	84

T = 0.4500000E 01				
0.527430E-01 0.105446				
H = 0.40960E 00	N5 =	2 N6 =	10 N4 =	85

T = 0.5000000E 01				
0.410677E-01 0.821201E-01				
H = 0.40960E 00	N5 =	2 N6 =	11 N4 =	86

T = 0.5500000E 01				
0.319846E-01 0.639539E-01				
H = 0.40960E 00	N5 =	2 N6 =	12 N4 =	87
H = 0.40960E 00	N5 =	2 N6 =	13 N4 =	88

T = 0.6000000E 01				
0.249095E-01 0.498066E-01				

H = 0.81920E 00 N5 = 5 N6 = 5 N4 = 89

T = 0.6500000E 01
0.194006E-01 0.387900E-01

T = 0.7000000E 01
0.151000E-01 0.301892E-01
H = 0.81920E 00 N5 = 4 N6 = 6 N4 = 90

T = 0.7500000E 01
0.117309E-01 0.234568E-01
H = 0.81920E 00 N5 = 2 N6 = 7 N4 = 91

T = 0.8000000E 01
0.911613E-02 0.182281E-01

T = 0.8500000E 01
0.708576E-02 0.141683E-01
H = 0.81920E 00 N5 = 2 N6 = 8 N4 = 92

T = 0.9000000E 01
0.550849E-02 0.110139E-01

T = 0.9500000E 01
0.428615E-02 0.856983E-02
H = 0.81920E 00 N5 = 2 N6 = 9 N4 = 93

T = 0.1000000E 02
0.333520E-02 0.666902E-02

STATE EQUATIONS EVALUATED 278 TIMES

JACOBIAN EVALUATED 4 TIMES